Code Smells Ana

Métodos diferentes com o mesmo nome (GanttDaysOff) GanttDaysOff.java

```
public boolean isADayOff(GanttCalendar date) {
 return (date.equals(myStart) || date.equals(myFinish) ||
(date.before(myFinish) && date.after(myStart)));
}

public boolean isADayOff(Date date) {
 return (date.equals(myStart.getTime()) ||
date.equals(myFinish.getTime()) ||
(date.before(myFinish.getTime()) &&
date.after(myStart.getTime())));
}
```

Métodos que recebem um parâmetro e não o usam (AlwaysWorkingTimeCalendarImpl) AlwaysWorkingTimeCalendarImpl.java

```
@Override
public DayType getWeekDayType(int day) {
 // Every day is a working day...
 return GPCalendar.DayType.WORKING;
}

@Override
public int getDayMask(Date date) {
 return GPCalendar.DayMask.WORKING;
}

@Override
public CalendarEvent getEvent(Date date) {
 return null;
}
```

Métodos que não fazem nada (AlwaysWorkingTimeCalendarImpl) AlwaysWorkingTimeCalendarImpl.java

```
@Override
public void setOnlyShowWeekends(boolean onlyShowWeekends) {
 // Ignore onlyShowWeekends, since weekends are always
 // working days for this calendar
}
```

```
@Override
public void setPublicHolidays(Collection<CalendarEvent>
holidays) {
}

@Override
public void setBaseCalendarID(String id) {
}

@Override
public void importCalendar(GPCalendar calendar,
ImportCalendarOption importOption) {
}
```

GoF Ana

Facade WeekendCalendarImpl.java

```
private Date getRecurringDate(Date date) {
 myCalendar.setTime(date);
 myCalendar.set(Calendar.YEAR,DUMMY_YEAR_FOR_RECURRING_EVENTS);
 return myCalendar.getTime();
}
```

Factory OffsetBuilderImpl.java

```
protected OffsetBuilderImpl(OffsetBuilder.Factory factory) {
 myCalendar = factory.myCalendar;
 myStartDate = factory.myStartDate;
 myViewportStartDate = factory.myViewportStartDate;
 myTopUnit = factory.myTopUnit;
 myBottomUnit = factory.myBottomUnit;
 myDefaultUnitWidth = factory.myAtomicUnitWidth;
 myChartWidth = factory.myEndOffset;
 myWeekendDecreaseFactor = factory.myWeekendDecreaseFactor;
 myEndDate = factory.myEndDate;
 baseUnit = factory.myBaseUnit;
 myRightMarginBottomUnitCount = factory.myRightMarginTimeUnits;
 myOffsetStepFn = factory.myOffsetStepFn;
}
```

Iterator TimelineSceneBuilder.java (Método renderTopUnits())

```
for (Offset nextOffset : topOffsets) {
 if (curX >= 0) {
   TimeUnitText[] texts =
myInputApi.getFormatter(nextOffset.getOffsetUnit(),
TimeUnitText.Position.UPPER_LINE)
       .format(nextOffset.getOffsetUnit(), curDate);
   final int maxWidth = nextOffset.getOffsetPixels() - curX - 5;
   final TimeUnitText timeUnitText = texts[0];
   textGroup.addText(curX + 5, 0, new TextSelector() {
     @Override
     public Canvas.Label[] getLabels(TextMetrics
textLengthCalculator) {
       return timeUnitText.getLabels(maxWidth,
textLengthCalculator);
     }
   });
   getTimelineContainer().createLine(curX, topUnitHeight - 10,
curX, topUnitHeight);
 }
 curX = nextOffset.getOffsetPixels();
 curDate = nextOffset.getOffsetEnd();
}
```

Code Smells João L.

Classe Inútil (WebStartIDClass)

WebStartIDClass.java

```
public class WebStartIDClass {

}
```

Método Longo (ProjectFileImporter)

ProjectFileImporter.java

```
private void importTask(Task t,
net.sourceforge.ganttproject.task.Task supertask,
                       Map<Integer, GanttTask>
foreignId2nativeTask, Map<GanttTask, Date>
nativeTask2foreignStart) {
 if (t.getNull()) {
   myErrors.add(Pair.create(Level.INFO,
       MessageFormat.format("Task with id={0} is blank task.
Skipped", foreignId(t))));
   return;
 }
 if (t.getUniqueID() == 0) {
   boolean isRealTask = t.getName() != null &&
!t.getChildTasks().isEmpty();
   if (!isRealTask) {
     for (Task child : t.getChildTasks()) {
       importTask(child, getTaskManager().getRootTask(),
foreignId2nativeTask, nativeTask2foreignStart);
     }
     return;
   }
 }

 StringBuilder report = new StringBuilder();
 java.util.function.Function<Task, Pair<TimeDuration,
TimeDuration>> getDuration = findDurationFunction(t, report);
 if (getDuration == null) {
   myErrors.add(Pair.create(Level.SEVERE,
       String.format("Can't determine the duration  of task %s
(%s). Skipped", t, report)));
   return;
 }

 TaskBuilder taskBuilder = getTaskManager().newTaskBuilder()
     .withParent(supertask)
     .withName(t.getName())
     .withNotes(t.getNotes())
     .withWebLink(t.getHyperlink());
 if (t.getPriority() != null) {
   taskBuilder =
taskBuilder.withPriority(convertPriority(t.getPriority()));
 }
 Date foreignStartDate = convertStartTime(t.getStart());
 if (t.getChildTasks().isEmpty()) {
   taskBuilder.withStartDate(foreignStartDate);
   if (t.getPercentageComplete() != null) {

taskBuilder.withCompletion(t.getPercentageComplete().intValue(
));
   }
   if (t.getMilestone()) {
     taskBuilder.withLegacyMilestone();
   }
   Pair<TimeDuration, TimeDuration> durations =
getDuration.apply(t);
```

```
   TimeDuration workingDuration = durations.first();
   TimeDuration nonWorkingDuration = durations.second();
   TimeDuration defaultDuration =
myNativeProject.getTaskManager().createLength(

myNativeProject.getTimeUnitStack().getDefaultTimeUnit(),
1.0f);

   if (!t.getMilestone()) {
     if (workingDuration.getLength() > 0) {
       taskBuilder.withDuration(workingDuration);
     } else if (nonWorkingDuration.getLength() > 0) {
       myErrors.add(Pair.create(Level.INFO,
MessageFormat.format(
           "[FYI] Task with id={0}, name={1}, start date={2},
end date={3}, milestone={4} has working time={5} and non
working time={6}.\n"
               + "We set its duration to {6}", foreignId(t),
t.getName(), t.getStart(), t.getFinish(),
           t.getMilestone(), workingDuration,
nonWorkingDuration)));
       taskBuilder.withDuration(nonWorkingDuration);
     } else {
       myErrors.add(Pair.create(Level.INFO,
MessageFormat.format(
           "[FYI] Task with id={0}, name={1}, start date={2},
end date={3}, milestone={4} has working time={5} and non
working time={6}.\n"
               + "We set its duration to default={7}",
foreignId(t), t.getName(), t.getStart(), t.getFinish(),
           t.getMilestone(), workingDuration,
nonWorkingDuration, defaultDuration)));
       taskBuilder.withDuration(defaultDuration);
     }
   } else {
     taskBuilder.withDuration(defaultDuration);
   }
  }
  GanttTask nativeTask = (GanttTask) taskBuilder.build();
  if (t.getCost() != null) {
   nativeTask.setCost(new
CostStub(BigDecimal.valueOf(t.getCost().doubleValue()),
false));
  }
```

```
 if (!t.getChildTasks().isEmpty()) {
   for (Task child : t.getChildTasks()) {
     importTask(child, nativeTask, foreignId2nativeTask,
nativeTask2foreignStart);
   }
 }
 importCustomFields(t, nativeTask);
 foreignId2nativeTask.put(foreignId(t), nativeTask);
 nativeTask2foreignStart.put(nativeTask, foreignStartDate);
}
```

Dead Code (ProjectFileExporter)
ProjectFileExporter.java

```
private void exportTasks(Map<Integer, net.sf.mpxj.Task>
id2mpxjTask) throws MPXJException {
//    Map<CustomPropertyDefinition, FieldType>
customProperty_fieldType = new
HashMap<CustomPropertyDefinition, FieldType>();
//
collectCustomProperties(getTaskManager().getCustomPropertyMana
ger(), customProperty_fieldType, TaskField.class);
   Map<CustomPropertyDefinition, FieldType>
customProperty_fieldType =
CustomPropertyMapping.buildMapping(getTaskManager());
   exportCustomFieldTypes(customProperty_fieldType);
   net.sf.mpxj.Task rootTask = myOutputProject.addTask();
   rootTask.setEffortDriven(false);
   rootTask.setID(0);
   rootTask.setUniqueID(0);
   rootTask.setOutlineLevel(0);
   rootTask.setWBS("0");
   rootTask.setOutlineNumber("0");

rootTask.setStart(convertStartTime(getTaskManager().getProject
Start()));

rootTask.setFinish(convertFinishTime(getTaskManager().getProje
ctEnd()));

rootTask.setDuration(convertDuration(getTaskManager().createLe
ngth(
getTaskManager().getRootTask().getDuration().getTimeUnit(),
getTaskManager().getProjectStart(),
       getTaskManager().getProjectEnd())));
   // rootTask.setDurationFormat(TimeUnit.DAYS);
   rootTask.setTaskMode(TaskMode.AUTO_SCHEDULED);

   int i = 0;
   for (Task t :
getTaskHierarchy().getNestedTasks(getTaskHierarchy().getRootTa
sk())) {
     exportTask(t, null, 1, ++i, id2mpxjTask,
customProperty_fieldType);
   }
 }
```

GoF João L.

Builder (IcsFileImporter)
Embora não tenha construtor tem métodos para mudar o objeto
[IcsFileImporter.java]

```
static class CalendarEditorPage implements WizardPage {
 private File myFile;
 private JPanel myPanel = new JPanel();
 private List<CalendarEvent> myEvents;
 private void setFile(File f) {
   myFile = f;
 }
 void setEvents(List<CalendarEvent> events) {
   myEvents = events;
 }
 List<CalendarEvent> getEvents() {
   return myEvents;
 }

 public String getTitle() {
   return
ourLocalizer.formatText("impex.ics.previewPage.title");
 }
 public JComponent getComponent() {
   return myPanel;
 }
 public void setActive(AbstractWizard wizard) {
   if (wizard != null) {
     myPanel.removeAll();
     if (myFile != null && myFile.exists() &&
myFile.canRead()) {
       if (myEvents != null) {
         myPanel.add(new
CalendarEditorPanel(wizard.getUIFacade(), myEvents,
null).createComponent());
         return;
       } else {
         LOGGER.error("No events found in file {}", new
Object[]{myFile}, Collections.emptyMap(), null);
       }
     } else {
       LOGGER.error("File {} is NOT readable", new
Object[]{myFile}, Collections.emptyMap(), null);
     }
     myPanel.add(new
JLabel(ourLocalizer.formatText("impex.ics.filePage.error.noEve
nts", myFile.getAbsolutePath())));
   }
 }
}
```

Facade (ImporterFromCsvFile)
[ImporterFromCsvFile.java]

```
public void run() {
 File selectedFile = getFile();
 BufferProject bufferProject = new BufferProject(getProject(),
getUiFacade());
 GanttCSVOpen opener = new GanttCSVOpen(selectedFile,
bufferProject.getTaskManager(),
     bufferProject.getHumanResourceManager(),
bufferProject.getRoleManager(),
     bufferProject.getTimeUnitStack());

opener.setOptions(((GanttProject)getProject()).getGanttOptions
().getCSVOptions());
 try {
   List<Pair<Level, String>> errors = opener.load();
   importBufferProject(getProject(), bufferProject,
BufferProjectImportKt.asImportBufferProjectApi(getUiFacade()),
       myMergeResourcesOption, null);
   reportErrors(errors, "CSV");
 } catch (Exception e) {
   getUiFacade().showErrorDialog(e);
 }
}
```

Proxy (DialogBuilder)
[DialogBuilder.java]

```
public Dialog createDialog(Component content, Action[]
buttonActions, String title, final NotificationManager
notificationManager) {
 final JDialog dlg = new JDialog(myMainFrame, true);
 final DialogImpl result = new DialogImpl(dlg, myMainFrame,
notificationManager);
 dlg.setTitle(title);
 dlg.getContentPane().setLayout(new BorderLayout());
 dlg.getContentPane().add(content, BorderLayout.CENTER);

 final Commiter commiter = new Commiter();
 Action cancelAction = null;
 int buttonCount = 0;
 if (buttonActions.length > 0) {
   JPanel buttonBox = new JPanel(new GridLayout(1,
buttonActions.length, 5, 0));
   for (final Action nextAction : buttonActions) {
     JButton nextButton = null;
     if (nextAction instanceof OkAction) {
       final JButton _btn = new JButton();
       final AbstractAction _delegate = (AbstractAction)
nextAction;
       OkAction proxy = new OkAction() {
         // These two steps handel the case when focus is
somewhere in text input
         // and user hits Ctrl+Enter
         // First we want to move focus to OK button to allow
focus listeners, if any,
         // to catch focusLost event
         // Second, we want it to happen before original
OkAction runs
```

```
         // So we wrap original OkAction into proxy which
moves focus and schedules "later" command
         // which call the original action. Between them EDT
sends out focusLost events.
         final Runnable myStep2 = new Runnable() {
           @Override
           public void run() {
             result.hide();
             commiter.commit();
             nextAction.actionPerformed(null);

_delegate.removePropertyChangeListener(myDelegateListener);
           }
         };
         final Runnable myStep1 = new Runnable() {
           @Override
           public void run() {
             _btn.requestFocus();
             SwingUtilities.invokeLater(myStep2);
           }
         };

         @Override
         public void actionPerformed(final ActionEvent e) {
           SwingUtilities.invokeLater(myStep1);
         }

         private void copyValues() {
           for (Object key : _delegate.getKeys()) {
             putValue(key.toString(),
_delegate.getValue(key.toString()));
           }
           setEnabled(_delegate.isEnabled());
         }

         private PropertyChangeListener myDelegateListener =
new PropertyChangeListener() {
           @Override
           public void propertyChange(PropertyChangeEvent evt)
{
             copyValues();
           }
         };
         {

_delegate.addPropertyChangeListener(myDelegateListener);
           copyValues();
         }
       };
       _btn.setAction(proxy);
       nextButton = _btn;

       if (((OkAction) nextAction).isDefault()) {
         dlg.getRootPane().setDefaultButton(nextButton);
       }

     }
```

Code Smells Rodrigo

Comentários como lembrete (CustomPropertyMapping)
CustomPropertyMapping.java

```
private void run0(Function<CustomPropertyDefinition, S>
fxnTaskField) {
        for (Iterator<CustomPropertyDefinition> it =
allDefs.iterator(); it.hasNext(); ) {
          CustomPropertyDefinition def = it.next();
            try {
              FieldType tf = fxnTaskField.apply(def);
              if (tf != null) {
                result.put(def, tf);
                mpxjFields.remove(tf);
                it.remove();
              }
            } catch (IllegalArgumentException e) {
              // That's somewhat okay. We have not found such
value in the enum, but it might come from the future
              // versions of MPXJ, so it is not the reason to
fail
            }
          }
      }
```

Método com parâmetro não utilizado (curDate) BottomUnitSceneBuilder.java

```
private void renderLabel(TextGroup textGroup, int curX, Date
curDate, Offset curOffset, TimeFormatter formatter) {
    final int maxWidth = curOffset.getOffsetPixels() - curX;
    TimeUnitText[] texts = formatter.format(curOffset);
    for (int i = 0; i < texts.length; i++) {
      final TimeUnitText timeUnitText = texts[i];
      textGroup.addText(curX + 2, i, new TextSelector() {
        @Override
        public Canvas.Label[] getLabels(TextMetrics
textLengthCalculator) {
          return timeUnitText.getLabels(maxWidth,
textLengthCalculator);
        }
      });
    }
  }
```

Dead code (WeekendCalendarImpl) WeekendCalendarImpl.java

```
@Override

  public void setPublicHolidays(Collection<CalendarEvent>
holidays) {

(...)

//    myCalendarUrl = calendarUrl;

//    clearPublicHolidays();

//    if (calendarUrl != null) {

//      XMLCalendarOpen opener = new XMLCalendarOpen();

//

//      HolidayTagHandler tagHandler = new
HolidayTagHandler(this);

//

//      opener.addTagHandler(tagHandler);

//      opener.addParsingListener(tagHandler);

//      try {

//        opener.load(calendarUrl.openStream());
//       } catch (Exception e) {

//         throw new RuntimeException(e);

//       }

//     }

  }
```

GoF Rodrigo

Iterator (ExporterToMsProjectFile.java)

```
private String getSelectedFormatExtension() {
    for (int i = 0; i < FILE_FORMAT_IDS.length; i++) {
      if (myFileFormat.equals(FILE_FORMAT_IDS[i])) {
        return FILE_EXTENSIONS[i];
      }
    }
    throw new IllegalStateException("Selected format=" +
myFileFormat + " has not been found in known formats:"
        + Arrays.asList(FILE_FORMAT_IDS));
  }
```

Facade (ProjectFileExporter.java)

```
private Date convertFinishTime(Date gpFinishDate) {
    Calendar c = (Calendar) Calendar.getInstance().clone();
    c.setTime(gpFinishDate);
    c.add(Calendar.DAY_OF_YEAR, -1);
    Date finishTime =
myOutputProject.getDefaultCalendar().getFinishTime(c.getTime()
);
    if (finishTime != null) {
         c.set(Calendar.HOUR, finishTime.getHours());
         c.set(Calendar.MINUTE, finishTime.getMinutes());
    }
    return c.getTime();

  }
```

Builder (Canvas.java)

```
public static class Shape {
    private Color myBackgroundColor;

    private Color myForegroundColor;

    private String myStyleName;

    private Object myModelObject;

    private boolean isVisible = true;

    private LinkedHashSet<String> myStyles;

    private Float myOpacity = null;

    private final Map<String, String> attributes = new
HashMap<>();

    private LinkedHashSet<String> getStyles() {
      if (myStyles == null) {
        myStyles = new LinkedHashSet<String>();
      }
      return myStyles;
    }

    public void addStyle(String style) {
      getStyles().add(style);
    }

    public boolean hasStyle(String style) {
      return getStyles().contains(style);
    }

    public void setStyle(String styleName) {
      myStyleName = styleName;
    }

    public String getStyle() {
      return myStyleName;
    }

    public Color getBackgroundColor() {
      return myBackgroundColor;
    }
```

```
  public void setBackgroundColor(Color myBackgroundColor) {
    this.myBackgroundColor = myBackgroundColor;
  }

  public Color getForegroundColor() {
    return myForegroundColor;
  }

  public void setForegroundColor(Color myForegroundColor) {
    this.myForegroundColor = myForegroundColor;
  }

  public Object getModelObject() {
    return myModelObject;
  }

  public void setModelObject(Object modelObject) {
    myModelObject = modelObject;
  }

  public boolean isVisible() {
    return isVisible;
  }

  public void setVisible(boolean visible) {
    isVisible = visible;
  }

  public Float getOpacity() {
    return myOpacity;
  }

  public void setOpacity(float opacity) {
    myOpacity = opacity;
  }

  public Map<String, String> getAttributes() {
    return attributes;
  }
}
```

Code Smells João A.

Cadeias de mensagem longas (long message chains) (ImporterFromMsProjectFile)

Este código possui cadeias de mensagem longas o que pode tornar o código pouco intuitivo e pouco prático por conseguinte.

Para melhorar este código podemos utilizar a lei de demeter para simplificar as cadeias de mensagem tornando o código mais inteligível.

ImporterFromMsProjectFile.java

```
public void run() {

    try {

      File selectedFile = getFile();

      BufferProject bufferProject = new
BufferProject(getProject(), getUiFacade());

      ProjectFileImporter importer = new
ProjectFileImporter(bufferProject,
getUiFacade().getTaskColumnList(), selectedFile);

      importer.run();

      List<Pair<Level, String>> errors = importer.getErrors();


getTaskManager().getAlgorithmCollection().getRecalculateTaskSc
heduleAlgorithm().setEnabled(false);


getTaskManager().getAlgorithmCollection().getRecalculateTaskCo
mpletionPercentageAlgorithm().setEnabled(false);


getTaskManager().getAlgorithmCollection().getScheduler().setEn
abled(false);

     …
```

Comentários com dead code (GanttCalendar)

Este método está comentado o que cria confusão no código para alguém que veja o código ou mesmo para alguém que esteja a trabalhar no código.

Neste caso, a melhor opção será ver se este código comentado é de relevância se for tirar os comentários, se não for retirá-lo do script.

GanttCalendar.java

```
//  /** @return the actually date */

//  public static String getDateAndTime() {

//    GanttCalendar c = new GanttCalendar();

//    return c.toString() + " - " +
GanttLanguage.getInstance().formatTime(c);

//  }
```

Métodos que não usam parâmetros (TaskDefaultColumn)

Estes métodos não utilizam os parâmetros que nos são dados logo criando um code smell.

Neste caso ou modifica-se os métodos para usarem os parâmetros ou alternativamente só tirar os parâmetros.

TaskDefaultColumn.java

```
static class Functions {

    static Predicate<Object> NOT_EDITABLE = new
Predicate<Object>() {

      @Override

      public boolean apply(Object input) {

        return false;

      }

    };
    static Predicate<Object> ALWAYS_EDITABLE = new
Predicate<Object>() {
      @Override
      public boolean apply(Object input) {
        return true;
      }
    };
```

GoF João A.

Builder (GPCalanderBase.java)

A classe não possui um construtor para inicializar as variáveis em vez disso depende de métodos para tal.

```
abstract class GPCalendarBase implements GPCalendarCalc {
  private final List<GPCalendarListener> myListeners =
Lists.newArrayList();
  private String myName;
 private String myId;
  @Override
  public String getID() {
    return myId == null ? myName : myId;
  }
  @Override
  public String getName() {
    return myName;
  }
  @Override
  public void setName(String name) {
    myName = name;
  }
  @Override
  public void setID(String id) {
    myId = id;
  }
```

Facade (TimeUnitImpl.java)

A classe possui variáveis que representam instâncias de partes mais complexas dos sistemas, assim simplificando o seu uso.

```
public class TimeUnitImpl implements TimeUnit {
  private final String myName;
  private final TimeUnitGraph myGraph;
  private final TimeUnit myDirectAtomUnit;
  public TimeUnitImpl(String name, TimeUnitGraph graph,
TimeUnit directAtomUnit) {
    myName = name;
    myGraph = graph;
    myDirectAtomUnit = directAtomUnit;
  }
```

State (WeekendCalendarImpl.java)

O método comporta-se de maneira diferente de acordo com o estado(que neste caso é o tipo de dia).

```
private boolean isPublicHoliDay(Date curDayStart) {
    CalendarEvent oneOff = myOneOffEvents.get(curDayStart);
    if (oneOff != null) {
      switch (oneOff.getType()) {
      case HOLIDAY:
        return true;

      case WORKING_DAY:

        return false;

      case NEUTRAL:

      default:

        // intentionally fall-through, consult recurring
holidays in this case

      }

    }

    CalendarEvent recurring =
myRecurringEvents.get(getRecurringDate(curDayStart));

    if (recurring != null) {

      switch (recurring.getType()) {

      case HOLIDAY:

        return true;

      case WORKING_DAY:

        return false;

      case NEUTRAL:

      default:

        // intentionally fall-through, use default answer

      }

    }

    return false;

  }
```

Code smells Ricardo Pereira

- Not using try-catch mechanism for error-checking (DesktopAdapter) DesktopAdapter.java

```
 @Override

      public void openFiles(OpenFilesEvent e) {

        List<File> files = e.getFiles();

        if (files.isEmpty()) {

          return;

        }

        File file = files.get(0);

        if (!file.isFile() || !file.canRead()) {

          return;

        }

        api.openFile(file);

      }
```

——

- No comments explaining the code (ExporterToHTML) ExporterToHTML.java

  (Whole class has no comments explaining the code, even on long, obscure methods)

```
private ExporterJob createGenerateGanttChartJob(final File outputFile, final List<File> resultFiles) {

    ExporterJob result = new ExporterJob("generate gantt chart") {

      @Override

      protected IStatus run() {

        try {

          int zoomLevel = getPreferences().getInt("zoom", -1);

          var exportSettings = createExportSettings();

          RenderedImage ganttChartImage = getGanttChart().asPrintChartApi().exportChart(

              exportSettings.getStartDate(), exportSettings.getEndDate(), zoomLevel,
exportSettings.isCommandLineMode());

          File ganttChartImageFile;

          ganttChartImageFile = replaceExtension(outputFile, GANTT_CHART_FILE_EXTENSION);

          ImageIO.write(ganttChartImage, PNG_FORMAT_NAME, ganttChartImageFile);
          resultFiles.add(ganttChartImageFile);

        } catch (IOException e) {

          getUIFacade().showErrorDialog(e);

          return Status.CANCEL_STATUS;

        } catch (OutOfMemoryError e) {

          getUIFacade().showErrorDialog(new RuntimeException("Out of memory when creating Gantt chart
image", e));

          return Status.CANCEL_STATUS;

        }

        return Status.OK_STATUS;

      }

    };

    return result;

}



  private ExporterJob createGenerateResourceChartJob(final File outputFile, final List<File>
resultFiles) {

    ExporterJob result = new ExporterJob("Generate resource chart") {

      @Override

      protected IStatus run() {

        try {

          int zoomLevel = getPreferences().getInt("zoom", -1);

          var exportSettings = createExportSettings();

          RenderedImage resourceChartImage = getResourceChart().asPrintChartApi().exportChart(

              exportSettings.getStartDate(), exportSettings.getEndDate(), zoomLevel,
exportSettings.isCommandLineMode());

          File resourceChartImageFile = replaceExtension(outputFile, RESOURCE_CHART_FILE_EXTENSION);

          ImageIO.write(resourceChartImage, PNG_FORMAT_NAME, resourceChartImageFile);

          resultFiles.add(resourceChartImageFile);

        } catch (IOException e) {

          getUIFacade().showErrorDialog(e);

          return Status.CANCEL_STATUS;
        } catch (OutOfMemoryError e) {

          getUIFacade().showErrorDialog(new RuntimeException("Out of memory when creating resource
chart image", e));

          return Status.CANCEL_STATUS;

        }

        return Status.OK_STATUS;

      }

    };

    return result;

  }
```

——

- Long parameter list (ExporterToHTML) ExporterToHTML

```
RenderedImage ganttChartImage = getGanttChart().asPrintChartApi().exportChart(

              exportSettings.getStartDate(), exportSettings.getEndDate(), zoomLevel,
exportSettings.isCommandLineMode());
```

GoF Ricardo Pereira

- **Abstract Factory** (StylesheetExporterBase) StylesheetExporterBase.java

(produce families of related objects without specifying their concrete classes)

```
public abstract class StylesheetExporterBase extends ExporterBase {

  private GPOptionGroup myOptions;

  protected EnumerationOption createStylesheetOption(String optionID, final List<Stylesheet>
stylesheets) {
  final List<String> names = new ArrayList<String>();

for (Stylesheet s : stylesheets) {

    names.add(s.getLocalizedName());

  }

  EnumerationOption stylesheetOption = new DefaultEnumerationOption<Stylesheet>(optionID, names) {

    @Override

    public void commit() {

      super.commit();

      String value = getValue();

      int index = names.indexOf(value);

      if (index >= 0) {

        setSelectedStylesheet(stylesheets.get(index));

      }

    }

  };

  return stylesheetOption;

}

@Override

public abstract String[] getFileExtensions();

protected abstract List<Stylesheet> getStylesheets();

protected abstract void setSelectedStylesheet(Stylesheet stylesheet);

protected abstract String getStylesheetOptionID();

public StylesheetExporterBase() {

}

@Override

public Component getCustomOptionsUI() {

  return null;

}

@Override

public void setContext(IGanttProject project, UIFacade uiFacade, Preferences prefs) {
  super.setContext(project, uiFacade, prefs);

  createStylesheetOption(getStylesheets());

}

private void createStylesheetOption(List<Stylesheet> stylesheets) {

  EnumerationOption stylesheetOption = createStylesheetOption(getStylesheetOptionID(), stylesheets);

  stylesheetOption.setValue(stylesheets.get(0).getLocalizedName());

  myOptions = new GPOptionGroup("exporter.html", new GPOption[] { stylesheetOption });

  myOptions.setTitled(false);

}

protected void setCommandLineStylesheet() {

  // Check if we are running from command line, if yes then we need to define the

  // stylesheet we are using

  if (getPreferences().getBoolean("commandLine", false) == true) {

    // Get the list of stylesheets

    List<Stylesheet> stylesheets = getStylesheets();



    // Set the first entry of list as default

    setSelectedStylesheet(stylesheets.get(0));



    // Test if a style is present in the arguments from command line

    // Iterate the list of style sheets to find it

    if (getPreferences().get("stylesheet", null) != null) {

      for (Stylesheet sheet : stylesheets) {

        if (sheet.getLocalizedName().compareTo(getPreferences().get("stylesheet", null)) == 0) {

          setSelectedStylesheet(sheet);

          break;

        }

      }

    }

  }
}

@Override

public GPOptionGroup getOptions() {

  return myOptions;

}

}
```

- **Builder** (ExporterToHTML) [ExporterToHTML.java]

    (doesn't implement a constructor, uses methods to define variables instead)

```
public class ExporterToHTML extends StylesheetExporterBase {

  static final String GANTT_CHART_FILE_EXTENSION = "png";

  static final String RESOURCE_CHART_FILE_EXTENSION = "res.png";
```

```
        private static final String PNG_FORMAT_NAME = "png";

        private HTMLStylesheet mySelectedStylesheet;

        private final HtmlSerializer mySerializer = new HtmlSerializer(this);

        @Override

        public String getFileTypeDescription() {

          return language.getText("impex.html.description");

        }

        @Override

        protected void setSelectedStylesheet(Stylesheet stylesheet) {

          mySelectedStylesheet = (HTMLStylesheet) stylesheet;

        }

       @Override

        public List<GPOptionGroup> getSecondaryOptions() {

          return null;

        }

        @Override

        public String getFileNamePattern() {

          return "html";

        }

      (...)
```

- **Singleton** (ImporterFromMsProjectFile) ImporterFromMsProjectFile.java

(Singleton is a creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance.)

```
private void findChangedDates(Map<GanttTask, Date> originalDates, Map<Task, Task> buffer2realTask,

      List<Pair<Level, String>> errors) {

   List<Pair<Level, String>> dateChangeMessages = Lists.newArrayList();
```

```
    for (Task bufferTask : originalDates.keySet()) {

      Date startPerMsProject = originalDates.get(bufferTask);

      if (startPerMsProject == null) {

        continue;

      }

      Task realTask = buffer2realTask.get(bufferTask);

      if (realTask == null) {

        continue;

      }

      Date startPerGanttProject = realTask.getStart().getTime();

      if (!startPerMsProject.equals(startPerGanttProject)) {

        dateChangeMessages.add(Pair.create(Level.WARNING, GanttLanguage.getInstance().formatText(

            "impex.msproject.warning.taskDateChanged", realTask.getName(), startPerMsProject,
startPerGanttProject)));

      }

    }

    if (!dateChangeMessages.isEmpty()) {

      errors.add(Pair.create(Level.INFO, GanttLanguage.getInstance().formatText(

          "impex.msproject.warning.taskDateChanged.heading", dateChangeMessages.size(),
originalDates.size())));

      errors.addAll(dateChangeMessages);

    }

  }
```